REST and RESTful APIs

What is REST?

REST stands for Representational State Transfer.
It is a design pattern used to create web services that provide a standard way for clients and servers to communicate over the internet using HTTP.
REST APIs are stateless, resource-based, and use standard HTTP methods like GET, POST, PUT, and DELETE.

RESTful Server vs RESTful API

- A RESTful server is a server that follows the REST architecture in how it handles requests and responses.
- A RESTful API is the actual interface (set of endpoints) exposed by the server for interaction.

Principles of REST

1. Separation of Clients and Server: Each can evolve independently.
2. Statelessness: The server does not retain any memory of the client between requests.
3. Access to Resources: Every resource is accessible through a unique URI.

API Resource Design - Bicycle Shop API

- Base URL: https://mikesbikes.com/api
- Endpoints:
  - /bikes (GET to retrieve all bikes, POST to add new)
  - /bikes/:id (GET, PUT, DELETE for a specific bike)
  - Query strings allow filtering like /bikes?type=mountain&brand=trek

Query Strings

- A way to filter results from an API.
- Always starts with '?' and uses '&' to chain filters.
  Example:
  /bikes?type=road&brand=trek
  Returns bikes where type = "road" and brand = "trek".

Common Questions Answered

1. How would you describe what REST is to your non-technical friend?

- → REST is like a set of rules websites use so different apps (like mobile or web) can talk to each other and share information easily.

2. What does a RESTful API usually return in response to incoming requests?

- → It usually returns structured data, most commonly in JSON format.

3. What kind of client devices can make use of a RESTful API?

- → Any device that can make HTTP requests — mobile apps, web apps, smart TVs, etc.

4. What does it mean for the server to be 'Stateless'?

- → It means the server forgets about any previous requests once it responds. Each request is treated independently.

Mike's Bikes API Example Tasks

2. Retrieve a list of all bikes that are sold

- → GET /bikes

3. Retrieve detailed info about bike with ID 42

- → GET /bikes/42

4. Update the price of the bike with ID 21

- → PUT /bikes/21

5. Add a new bike to the list of bikes being sold

- → POST /bikes

6. Remove the bike with ID 56 from the list

- → DELETE /bikes/56

Nested vs Non-Nested Resources

- /bikes/123/reviews: Reviews specifically for bike ID 123 (nested)
- /reviews: All reviews, regardless of which bike (non-nested)

What is a URL Parameter?

→ A placeholder in the URL path used to represent a specific resource, e.g., `/bikes/:id` where `:id` might be `123`

Bike Racks API – Expected Endpoints

1. Get a list of all bike racks sold on the site

- → /bikeracks?type=sold

2. Get a list of all bike racks available in the store

- → /bikeracks?available=true

3. Get a list of all 'Thule' bike racks that can hold 4 bikes

- → /bikeracks?brand=Thule&numBikes=4